We are using the BERT(Binary Encoder Representation of Transformers) for training a model. We will use the pre-trained text to text h5 model provided by the hugging face library. The model is trained on a large corpus of text and thius helps learn the word embeddings easily. We have choosen this model to get rid of the large and the long process of model training. Transfer learning is the new state-of-the-art technology and hence we are using a pre-trained model for that.

PART - 1

In this part we are going to load the pretrained model from the transformers library and do the required text preprocessing. After that we will get the predictions for all the articles provided to us in the test file and store them in submission file

```
%pip install transformers==2.8.0
import torch
import json
import csv
import numpy as np
import pandas as pd
from transformers import T5Tokenizer, T5ForConditionalGeneration, T5Config
```

```
Collecting transformers==2.8.0
  Downloading https://files.pythonhosted.org/packages/a3/78/92cedda05552398352ed97849
     |████████████████████████████████| 573kB 2.8MB/s
Collecting tokenizers==0.5.2
  Downloading https://files.pythonhosted.org/packages/d1/3f/73c881ea4723e43c1e9acf317
     |████████████████████████████████| 3.7MB 29kB/s
Collecting sentencepiece
  Downloading https://files.pythonhosted.org/packages/d4/a4/d0a884c4300004a78cca907a6
     |████████████████████████████████| 1.1MB 31.6MB/s
Requirement already satisfied: boto3 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: filelock in /usr/local/lib/python3.6/dist-packages (fr
```

Importing the pretrained model from the Hugging Face Transformers Library

```
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from
```

```
pretrained_model = T5ForConditionalGeneration.from_pretrained('t5-small')
```

```
Downloading: 100%                    1.20k/1.20k [00:00<00:00, 3.32kB/s]

Downloading: 100%                    242M/242M [00:16<00:00, 14.9MB/s]

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-pa
```

Importing and using the built in tokenizer provided by the Hugging Face Library

```
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from sa
```

```
tokenizer = T5Tokenizer.from_pretrained('t5-small', do_lower_case = False)
device = torch.device("cpu")
```

```
Downloading: 100%                    792k/792k [00:02<00:00, 322kB/s]

  Stored in directory: /root/.cache/pip/wheels/29/3c/fd/7ce5c3f0666dab31a50123635e6fb
```

Importing the dataset that we have

```
Successfully installed sacremoses-0.0.43 sentencepiece-0.1.91 tokenizers-0.5.2 transf
```

```
data = pd.read_excel("/content/TASK.xlsx", skiprows = 1)
print(data.shape)
print(data.head())
```

```
(1000, 2)
   Unnamed: 0                                       Introduction
0         NaN  Acnesol Gel is an antibiotic that fights bacte...
1         NaN  Ambrodil Syrup is used for treating various re...
2         NaN  Augmentin 625 Duo Tablet is a penicillin-type ...
3         NaN  Azithral 500 Tablet is an antibiotic used to t...
4         NaN  Alkasol Oral Solution is a medicine used in th...
```

The only relevant column in this dataset is the text that we want to summarize so we are keeping that only in out dataframe

```
data = data["Introduction"]
print(data.head())
```

```
0    Acnesol Gel is an antibiotic that fights bacte...
1    Ambrodil Syrup is used for treating various re...
2    Augmentin 625 Duo Tablet is a penicillin-type ...
3    Azithral 500 Tablet is an antibiotic used to t...
4    Alkasol Oral Solution is a medicine used in th...
Name: Introduction, dtype: object
```

```
def preprocess(text):
  text = text.strip().replace("\n", "")
  text = "summarize: " + text
  return text

for i in range(len(data)):
  data[i] =  preprocess(data[i])


summaries = []
```

Now its time to tokenize all the text using thee tokenizer provided to us.

```
def process(text):
  tokenized_text = tokenizer.encode(text, return_tensors="pt").to(device)
  summary_ids = pretrained_model.generate(tokenized_text,
                                    num_beams=4,
                                    no_repeat_ngram_size=2,
                                    min_length=50,
                                    max_length=100,
                                    early_stopping=True)
  output = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
  summaries.append(output)


for text in data[:10]:
  process(text)
```

...

Saving the summaries in the submission file

```
data = pd.read_excel("/content/TASK.xlsx", skiprows = 1)
data["Summary"] = summaries
data.to_csv("submission.csv", index = False)
```

PART-2

Now in this secction we are going to make the same function but this function will take a sentence from the user and the lengths of output required by the model.

```
def processSentence(text, min, max):
  tokenized_text = tokenizer.encode(text, return_tensors="pt").to(device)
  summary_ids = pretrained_model.generate(tokenized_text,
                             num_beams=4,
                             no_repeat_ngram_size=2,
                             min_length=min,
                             max_length=max,
                             early_stopping=True)
  output = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
  return output
```

Now we are required to take the input from the user and ask for the minimum andd the maximum length of the summary from the user and then do thee processing of the input to get the output

```
text = input("Enter the text: ")
min = input("Enter the minimum summary length: ")
max = input("Enter the maximum summary length: ")
text = preprocess(text)
output = processSentence(text, int(min), int(max))
print(output)
```